MCA 304A DATA WAREHOUSING AND DATA MINING

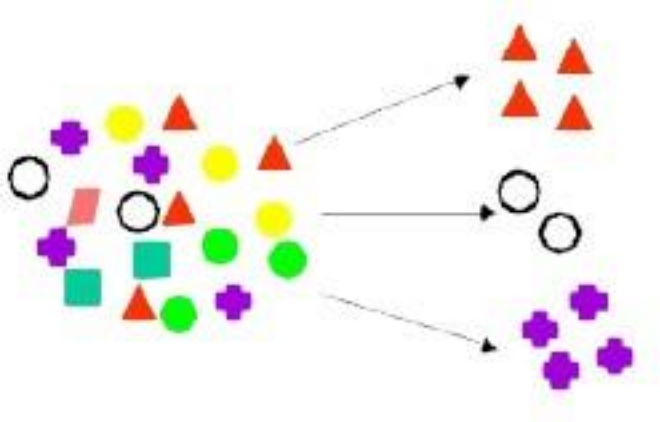
UNIT V

Clustering Techniques

.

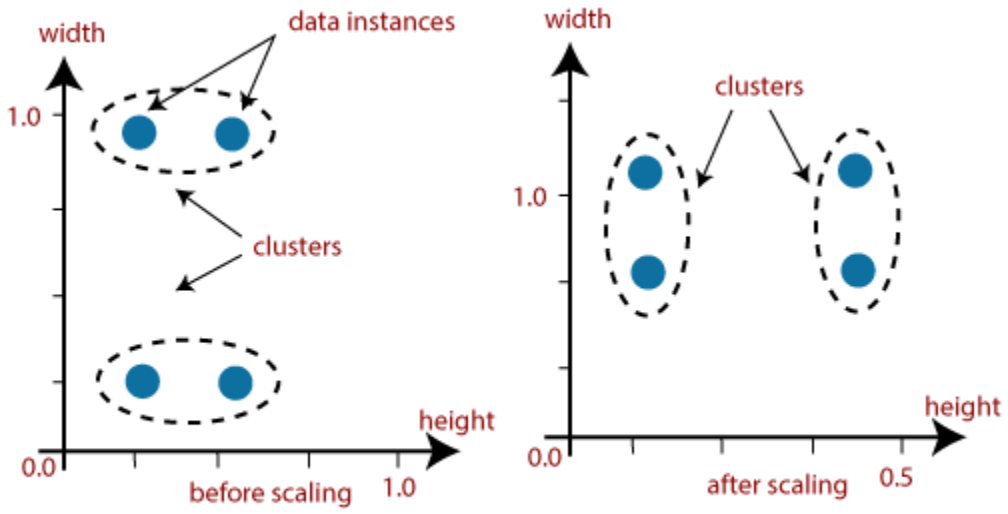
Prepared by S NOORTAJ Asst prof.

Cluster Analysis in Data Mining means that to find out the group of objects which are similar to each other in the group but are different from the object in other groups. In the process of clustering in data analytics, the sets of data are divided into groups or classes based on data similarity. Then each of these classes is labelled according to their data types.



The main **features** with the data clustering algorithms is that it cant be standardized.

1. Scalability: Scalability in clustering implies that as we boost the amount of data objects, the time to perform clustering should approximately scale to the complexity order of the algorithm. For example, if we perform K- means clustering, we know it is O(n), where n is the number of objects in the data. If we raise the number of data objects 10 folds, then the time taken to cluster them should also approximately increase 10 times. It means there should be a linear relationship. If that is not the case, then there is some error with our implementation process.



Showing example where scalability may leads to wrong result

2. Interpretability:

The outcomes of clustering should be interpretable, comprehensible, and usable.

3. Discovery of clusters with attribute shape:

The clustering algorithm should be able to find arbitrary shape clusters. They should not be limited to only distance measurements that tend to discover a spherical cluster of small sizes.

4. Ability to deal with different types of attributes:

Algorithms should be capable of being applied to any data such as data based on intervals (numeric), binary data, and categorical data.

5. Ability to deal with noisy data:

Databases contain data that is noisy, missing, or incorrect. Few algorithms are sensitive to such data and may result in poor quality clusters.

6. High dimensionality:

The clustering tools should not only able to handle high dimensional data space but also the low-dimensional space.

Types of Data and Computing Distance

A distance measure, dis(ti, tj), as opposed to similarity, is often used in clustering. The clustenng problem then has the desirable property that given a cluster Kj. For all tjl, tjm E Kj and ti not belongs to Kj, dis(tjl, tjm) <=dis(tjl, ti).

$$\text{centroid} = C_m = \frac{\sum_{i=1}^{N}(t_{mi})}{N}$$

$$\text{radius} = R_m = \sqrt{\frac{\sum_{i=1}^{N}(t_{mi} - C_m)^2}{N}}$$

$$\text{diameter} = D_m = \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}(t_{mi} - t_{mj})^2}{(N)(N-1)}}$$

Many clustering algorithms require that the distance between clusters (rather than elements) be determined. This is not an easy task given that there are many interpretations for distance between clusters. Given clusters Ki and KJ, there are several standard alternatives to calculate the distance between clusters. A representative list is:

- **Single link:** Smallest distance between an element in one cluster and an element in the other. We thus have $\text{dis}(K_i, K_j) = \min(\text{dis}(t_{il}, t_{jm}))\forall t_{il} \in K_i \notin K_j$ and $\forall t_{jm} \in K_j \notin K_i$.
- **Complete link:** Largest distance between an element in one cluster and an element in the other. We thus have $\text{dis}(K_i, K_j) = \max(\text{dis}(t_{il}, t_{jm}))\forall t_{il} \in K_i \notin K_j$ and $\forall t_{jm} \in K_j \notin K_i$.
- **Average:** Average distance between an element in one cluster and an element in the other. We thus have $\text{dis}(K_i, K_j) = \text{mean}(\text{dis}(t_{il}, t_{jm}))\forall t_{il} \in K_i \notin K_j$ and $\forall t_{jm} \in K_j \notin K_i$.
- **Centroid:** If clusters have a representative centroid, then the centroid distance is defined as the distance between the centroids. We thus have $\text{dis}(K_i, K_j) = \text{dis}(C_i, C_j)$, where C_i is the centroid for K_i and similarly for C_j.
- **Medoid:** Using a medoid to represent each cluster, the distance between the clusters can be defined by the distance between the medoids: $\text{dis}(K_i, K_j) = \text{dis}(M_i, M_j)$.

Different types of clustering in data mining:

- Hierarchical Clustering Methods. ...
- Partitioning Clustering Method. ...
- Density-Based Clustering Method. ...

Hierarchical clustering methods

Hierarchical clustering refers to an unsupervised learning procedure that determines successive clusters based on previously defined clusters. It works via grouping data into a tree of clusters. Hierarchical clustering stats by treating each data points as an individual cluster. The endpoint refers to a different set of clusters, where each cluster is different from the other cluster, and the objects within each cluster are the same as one another. There are two types of hierarchical clustering

- Agglomerative Hierarchical Clustering
- Divisive Clustering

Agglomerative hierarchical clustering

Agglomerative clustering is one of the most common types of hierarchical clustering used to group similar objects in clusters. Agglomerative clustering is also known as AGNES (Agglomerative Nesting). In agglomerative clustering, each data point act as an individual cluster and at each step, data objects are grouped in a bottom-up method. Initially, each data object is in its cluster. At each iteration, the clusters are combined with different clusters until one cluster is formed.

Agglomerative hierarchical clustering algorithm

1. Determine the similarity between individuals and all other clusters. (Find proximity matrix).
2. Consider each data point as an individual cluster.
3. Combine similar clusters.
4. Recalculate the proximity matrix for each cluster.
5. Repeat step 3 and step 4 until you get a single cluster.

Let's understand this concept with the help of graphical representation using a dendrogram.

With the help of given demonstration, we can understand that how the actual algorithm work. Here no calculation has been done below all the proximity among the clusters are assumed.

Let's suppose we have six different data points P, Q, R, S, T, V.

Step 1:

Consider each alphabet (P, Q, R, S, T, V) as an individual cluster and find the distance between the individual cluster from all other clusters.

Step 2:

Now, merge the comparable clusters in a single cluster. Let’s say cluster Q and Cluster R are similar to each other so that we can merge them in the second step. Finally, we get the clusters [(P), (QR), (ST), (V)]

Step 3:

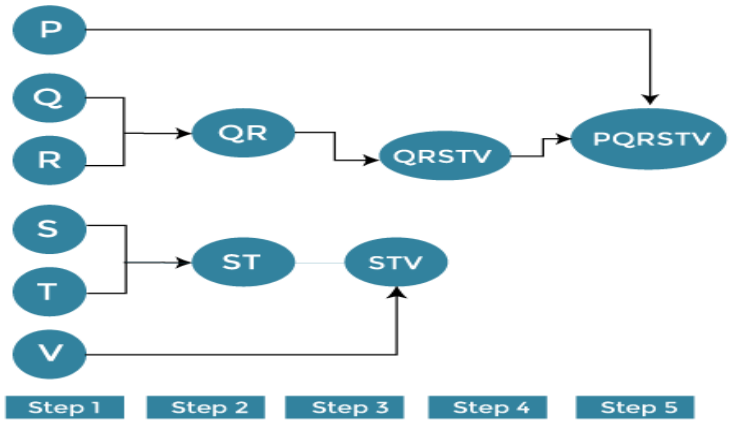
Here, we recalculate the proximity as per the algorithm and combine the two closest clusters [(ST), (V)] together to form new clusters as [(P), (QR), (STV)]

Step 4:

Repeat the same process. The clusters STV and PQ are comparable and combined together to form a new cluster. Now we have [(P), (QQRSTV)].

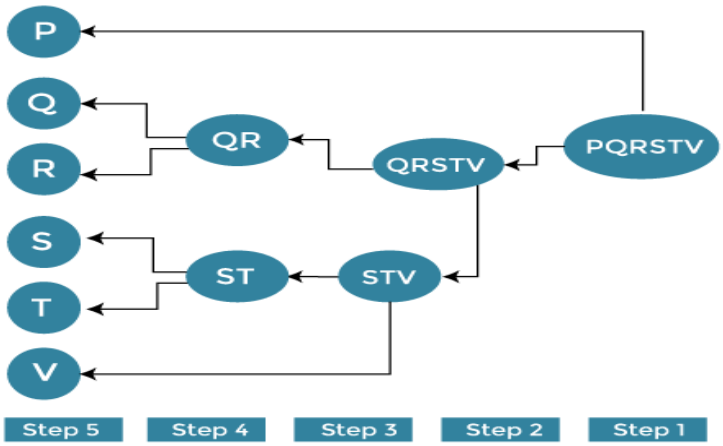
Step 5:

Finally, the remaining two clusters are merged together to form a single cluster [(PQRSTV)]



Divisive Hierarchical Clustering

Divisive hierarchical clustering is exactly the opposite of Agglomerative Hierarchical clustering. In Divisive Hierarchical clustering, all the data points are considered an individual cluster, and in every iteration, the data points that are not similar are separated from the cluster. The separated data points are treated as an individual cluster. Finally, we are left with N clusters.



Advantages of Hierarchical clustering

- It is simple to implement and gives the best output in some cases.
- It is easy and results in a hierarchy, a structure that contains more information.
- It does not need us to pre-specify the number of clusters.

Disadvantages of hierarchical clustering

- It breaks the large clusters.
- It is Difficult to handle different sized clusters and convex shapes.
- It is sensitive to noise and outliers.
- The algorithm can never be changed or deleted once it was done previously.

Partitional clustering methods:

K-Medoids and K-Means are two types of clustering mechanisms in Partition Clustering.

K-medoids is an unsupervised method with unlabelled data to be clustered. It is an improvised version of the K-Means algorithm mainly designed to deal with outlier data sensitivity. Compared to other partitioning algorithms, the algorithm is simple, fast, and easy to implement.

K-means clustering – K-means clustering is the most common partitioning algorithm. K-means reassigns each data in the dataset to only one of the new clusters formed. A record or data point is assigned to the nearest cluster using a measure of distance or similarity.

- ❖ It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.
- ❖ It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

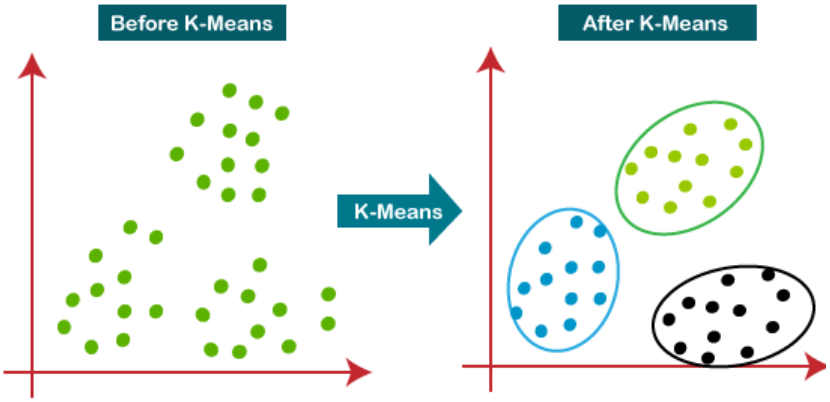
There are the following steps used in the K-means clustering

- It can select K initial cluster centroid c_1, c_2, c_3 … . c_k.
- It can assign each instance x in the S cluster whose centroid is nearest to x.
- For each cluster, recompute its centroid based on which elements are contained in that cluster.
- Go to (b) until convergence is completed.
- It can separate the object (data points) into K clusters.
- It is used to cluster center (centroid) = the average of all the data points in the cluster.
- It can assign each point to the cluster whose centroid is nearest (using distance function).

The initial values for the means are arbitrarily assigned. These can be assigned randomly or perhaps can use the values from the first k input items themselves.

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.



How does the K-Means Algorithm Work?

The working of the K-Means algorithm is explained in the below steps:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

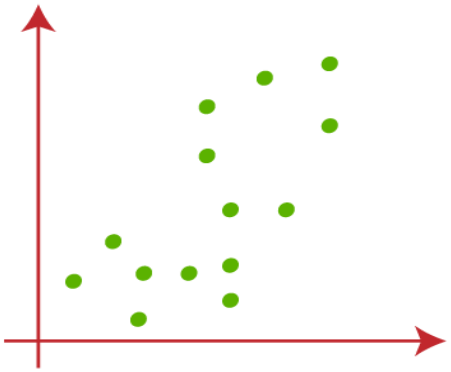
Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

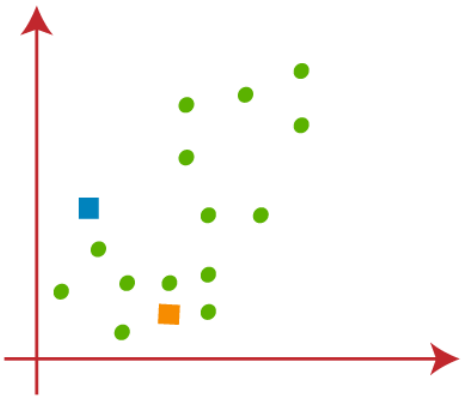
Step-7: The model is ready.

Let's understand the above steps by considering the visual plots:

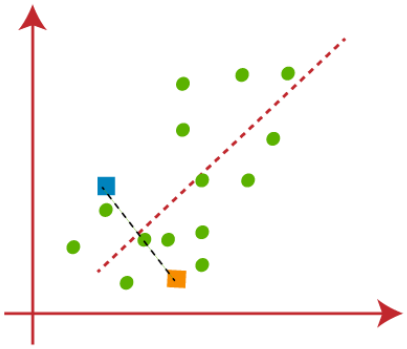
Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is given below:



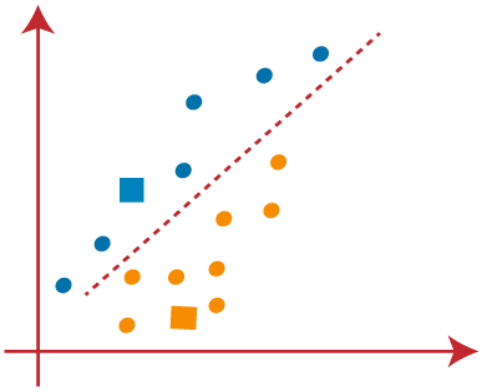
- Let's take number k of clusters, i.e., K=2, to identify the dataset and to put them into different clusters. It means here we will try to group these datasets into two different clusters.
- We need to choose some random k points or centroid to form the cluster. These points can be either the points from the dataset or any other point. So, here we are selecting the below two points as k points, which are not the part of our dataset. Consider the below image:



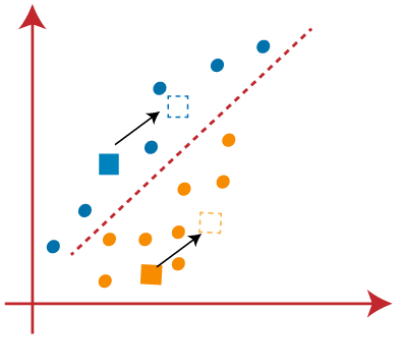
Now we will assign each data point of the scatter plot to its closest K-point or centroid. We will compute it by applying some mathematics that we have studied to calculate the distance between two points. So, we will draw a median between both the centroids. Consider the below image:



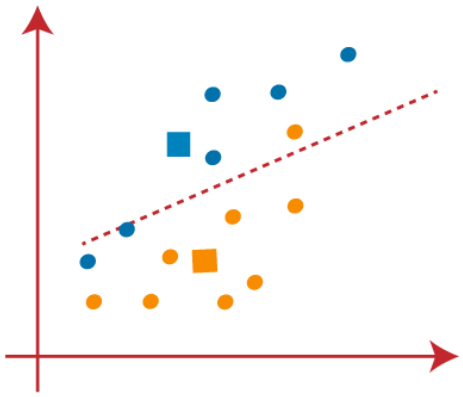
From the above image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid. Let's color them as blue and yellow for clear visualization.



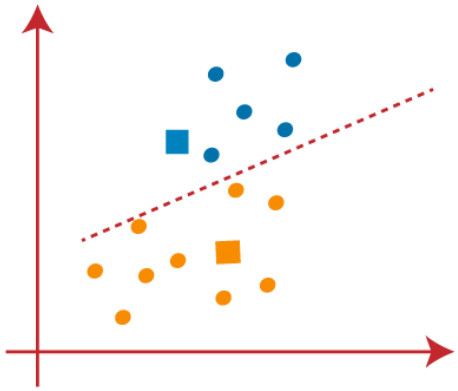
As we need to find the closest cluster, so we will repeat the process by choosing **a new centroid**. To choose the new centroids, we will compute the center of gravity of these centroids, and will find new centroids as below:



Next, we will reassign each datapoint to the new centroid. For this, we will repeat the same process of finding a median line. The median will be like below image:

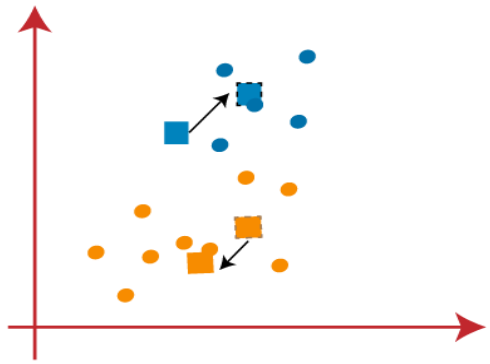


From the above image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.

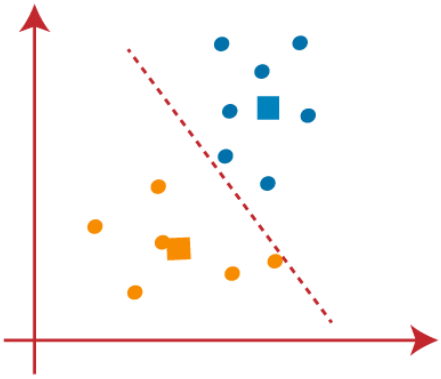


As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.

- We will repeat the process by finding the center of gravity of centroids, so the new centroids will be as shown in the below image:



As we got the new centroids so again will draw the median line and reassign the data points. So, the image will be:



K-Medoids

Medoid: A Medoid is a point in the cluster from which the sum of distances to other data points is minimal.

(or)

A Medoid is a point in the cluster from which dissimilarities with all the other points in the clusters are minimal.

Instead of centroids as reference points in K-Means algorithms, the K-Medoids algorithm takes a Medoid as a reference point.

Algorithm:

Given the value of k and unlabelled data:

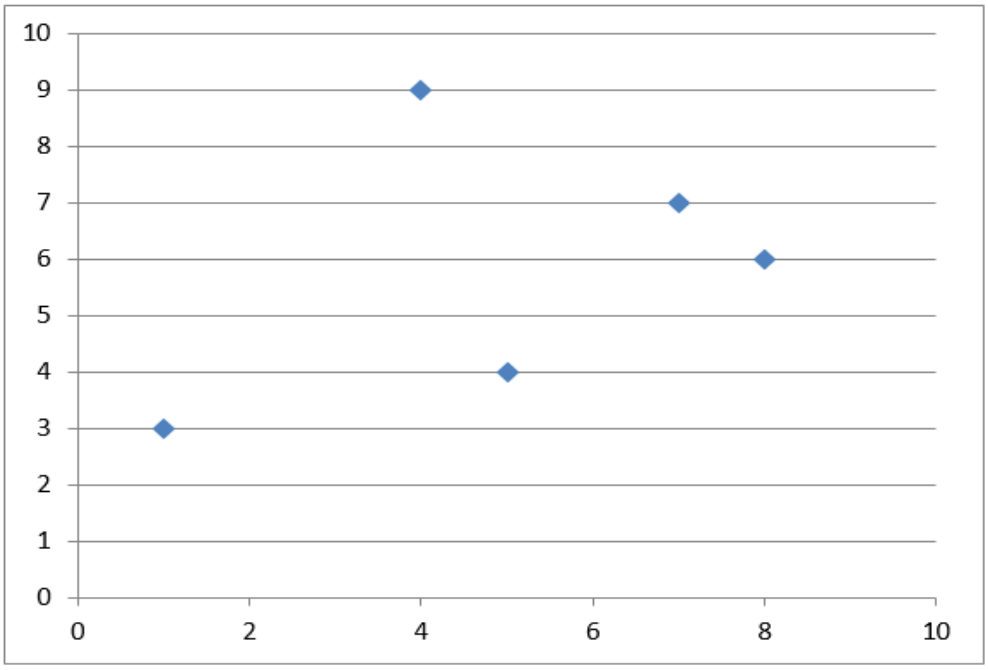
1. Choose k number of random points from the data and assign these k points to k number of clusters. These are the initial medoids.
2. For all the remaining data points, calculate the distance from each medoid and assign it to the cluster with the nearest medoid.
3. Calculate the total cost (Sum of all the distances from all the data points to the medoids)
4. Select a random point as the new medoid and swap it with the previous medoid. Repeat 2 and 3 steps.
5. If the total cost of the new medoid is less than that of the previous medoid, make the new medoid permanent and repeat step 4.
6. If the total cost of the new medoid is greater than the cost of the previous medoid, undo the swap and repeat step 4.
7. The Repetitions have to continue until no change is encountered with new medoids to classify data points.

Here is an example to make the theory clear:

Data set:

X	Y
5	4
7	7
1	3
8	6
4	9

Scatter plot:



If k is given as 2, we need to break down the data points into 2 clusters.

1. **Initial medoids: M1(1, 3) and M2(4, 9)**
2. Calculation of distances

Manhattan Distance: |x1 - x2| + |y1 - y2|

s. no.	X	y	From M1(1, 3)	From M2(4, 9)
0	5	4	5	6
1	7	7	10	5
2	1	3	-	-
3	8	6	10	7
4	4	9	-	-

Cluster 1: 0

Cluster 2: 1, 3

1. Calculation of total cost:
   (5) + (5 + 7) = 17
2. Random medoid: (5, 4)

M1(5, 4) and M2(4, 9):

s.no.	x	Y	From M1(5, 4)	From M2(4, 9)
0	5	4	-	-
1	7	7	5	5
2	1	3	5	9
3	8	6	5	7
4	4	9	-	-

Cluster 1: 2, 3

Cluster 2: 1

1. Calculation of total cost:
   (5 + 5) + 5 = 15
   Less than the previous cost
   New medoid: (5, 4).
2. Random medoid: (7, 7)

M1(5, 4) and M2(7, 7)

S. no	x	Y	From M1(5, 4)	From M2(7, 7)
0	5	4	-	-
1	7	7	-	-
2	1	3	5	10
3	8	6	5	2
4	4	9	6	5

Cluster 1: 2

Cluster 2: 3, 4

1. Calculation of total cost:
   (5) + (2 + 5) = 12
   Less than the previous cost
   New medoid: (7, 7).
2. Random medoid: (8, 6)

M1(7, 7) and M2(8, 6)

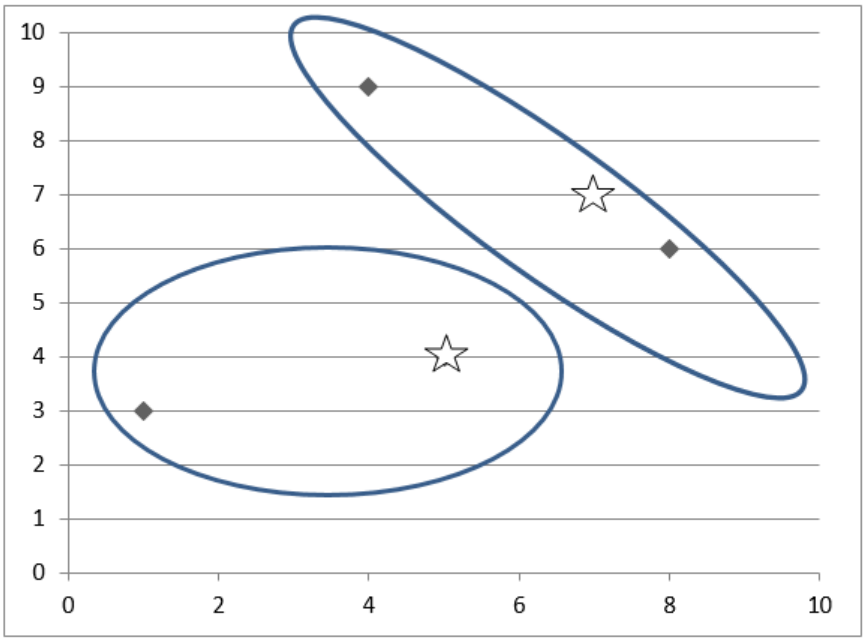
s. no.	X	Y	From M1(7, 7)	From M2(8, 6)
0	5	4	5	5
1	7	7	-	-
2	1	3	10	10
3	8	6	-	-

4	4	9	5	7

Cluster 1: 4

Cluster 2: 0, 2

1. Calculation of total cost:
   (5) + (5 + 10) = 20
   Greater than the previous cost
   UNDO
   Hence, the final medoids:
2. **M1(5, 4) and M2(7, 7)**
   Cluster 1: 2
   Cluster 2: 3, 4
   Total cost: 12
   Clusters:



Advantages of using K-Medoids:

1. Deals with noise and outlier data effectively
2. Easily implementable and simple to understand

3. Faster compared to other partitioning algorithms

Disadvantages:

1. Not suitable for Clustering arbitrarily shaped groups of data points.
2. As the initial medoids are chosen randomly, the results might vary based on the choice in different runs.

K-Means and K-Medoids:

K-Means	**K-Medoids**
Both methods are types of Partition Clustering.	
Unsupervised iterative algorithms	
Have to deal with unlabelled data	
Both algorithms group n objects into k clusters based on similar traits where k is pre-defined.	
Inputs: Unlabelled data and the value of k	
Metric of similarity: Euclidian Distance	**Metric of similarity:** Manhattan Distance
Clustering is done based on distance from **centroids.**	Clustering is done based on distance from **medoids.**
A centroid can be a data point or some other point in the cluster	A medoid is always a data point in the cluster.
Can't cope with outlier data	Can manage outlier data too
Sometimes, outlier sensitivity can turn out to be useful	Tendency to ignore meaningful clusters in outlier data

Density Based clustering Methods

Partitioning and hierarchical methods are designed to find spherical-shaped clusters.

They have difficulty finding clusters of arbitrary shape such as the “S” shape and oval

clusters in below figure.

Clusters of arbitrary shape.

Given such data, they would likely inaccurately identify convex regions, where noise or outliers are included in the clusters.

- ❖ To find clusters of arbitrary shape, alternatively, we can model clusters as dense regions in the data space, separated by sparse regions.
- ❖ This is the main strategy behind *density-based clustering methods,* which can discover clusters of nonspherical shape.
- ❖ The basic techniques of density-based clustering by studying three representative methods, namely,

- DBSCAN,
- OPTICS,
- DENCLUE.

DBSCAN: Density-Based Clustering Based on Connected Regions with High Density

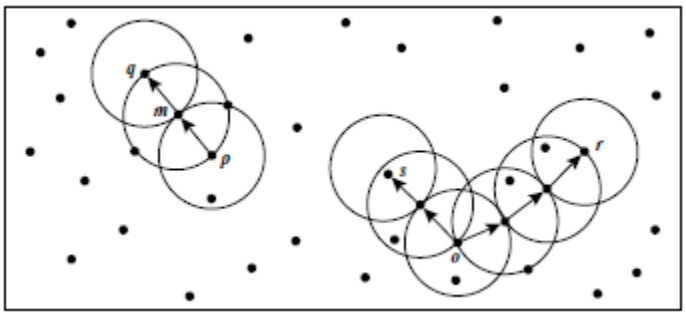
The *density* of an object ***o*** can be measured by the number of objects close to ***o***. **DBSCAN** (Density-Based Spatial Clustering of Applications with Noise) finds *core objects*, that is, objects that have dense neighborhoods. It connects core objects and their neighborhoods to form dense regions as clusters.

- ❖ A user-specified parameter E_ > 0 is used to specify the radius of a neighborhood we consider for every object.The _**-neighborhood** of an object ***o*** is the space within a radius _ centered at ***o***.
- ❖ the **density of a neighborhood** can be measured simply by the number of objects in the neighborhood.

- DBSCAN uses another user-specified parameter, ***MinPts,*** which specifies the density threshold of dense regions. An object is a **core object** if the _-neighborhood of the object contains at least *MinPts* objects.
- Core objects are the pillars of dense regions. To connect core objects as well as their neighbors in a dense region, **DBSCAN** uses the notion of density-connectedness. Two objects ***p*1**,***p*2** 2 *D* are **density-connected** with respect to _ and *MinPts* if there is an object ***q*** 2 *D* such that both ***p*1** and ***p*2** are density reachable from ***q*** with respect to _ and *MinPts*.
- Unlike density-reach ability, density connectedness is an equivalence relation. It is easy to show that, for objects ***o*1**, ***o*2**, and ***o*3**, if ***o*1** and ***o*2** are density-connected, and ***o*2** and ***o*3** are density-connected, then so are ***o*1** and ***o*3**.

Density-reachability and density-connectivity:

Consider the below figure for a given E represented by the radius of the circles, and, say, let *MinPts* =3.
the closure of density-connectedness to find connected dense regions as clusters. Each closed set is a density-based cluster.



the labeled points, ***m***,***p***,***o***, ***r*** are core objects because each is in an E-neighborhood containing at least three points. Object ***q*** is directly density-reachable from ***m***. Object ***m*** is directly density-reachable from ***p*** and vice versa. Object ***q*** is (indirectly) density-reachable from ***p*** because ***q*** is directly density reachable from ***m*** and ***m*** is directly density-reachable from ***p***. However, ***p*** is not density reachable from ***q*** because ***q*** is not a core object. Similarly, ***r*** and ***s*** are density-reachable from ***o*** and ***o*** is density-reachable from ***r***. Thus, ***o***, ***r***, and ***s*** are all density-connected.

"How does DBSCAN find clusters?"

- Initially, all objects in a given data set *D* are marked as "unvisited." DBSCAN randomly selects an unvisited object ***p***, marks ***p*** as "visited," and checks whether the E_-neighborhood of ***p*** contains at least *MinPts* objects.
- If not, ***p*** is marked as a noise point. Otherwise, a new cluster *C* is created for ***p***, and all the objects in the E_-neighborhood of ***p*** are added to a candidate set, *N*.

- DBSCAN iteratively adds to *C* those objects in *N* that do not belong to any cluster.
- In this process, for an object ***p***0 in *N* that carries the label "unvisited," DBSCAN marks it as "visited" and checks its E _-neighborhood. If the E_-neighborhood of ***p***0 has at least *MinPts* objects, those objects in the E_-neighborhood of ***p***0 are added to *N*.
- DBSCAN continues adding objects to *C* until *C* can no longer be expanded, that is, *N* is empty. At this time, cluster *C* is completed, and thus is output.

the computational complexity of DBSCAN is *O.n*log*n/*, where *n* is the number of database objects. Otherwise, the complexity is *O.n*2*/*. With appropriate settings of the user-defined parameters, E_ and *MinPts*, the algorithm is effective in finding arbitrary-shaped clusters.

To find the next cluster, DBSCAN randomly selects an unvisited object from the remaining ones. The clustering process continues until all objects are visited. The pseudocode of the DBSCAN algorithm is given in below

Algorithm: DBSCAN: a density-based clustering algorithm.

Input:

- D: a data set containing n objects,
- ϵ: the radius parameter, and
- *MinPts*: the neighborhood density threshold.

Output: A set of density-based clusters.

Method:

```
(1)  mark all objects as unvisited;
(2)  do
(3)      randomly select an unvisited object p;
(4)      mark p as visited;
(5)      if the ε-neighborhood of p has at least MinPts objects
(6)          create a new cluster C, and add p to C;
(7)          let N be the set of objects in the ε-neighborhood of p;
(8)          for each point p' in N
(9)              if p' is unvisited
(10)                 mark p' as visited;
(11)                 if the ε-neighborhood of p' has at least MinPts points,
                     add those points to N;
(12)             if p' is not yet a member of any cluster, add p' to C;
(13)         end for
(14)         output C;
(15)     else mark p as noise;
(16) until no object is unvisited;
```

OPTICS: Ordering Points to Identify the Clustering Structure

To overcome the difficulty in using one set of global parameters in clustering analysis, a cluster analysis method called OPTICS was proposed. OPTICS does not explicitly produce a data set clustering. Instead, it outputs a cluster ordering.

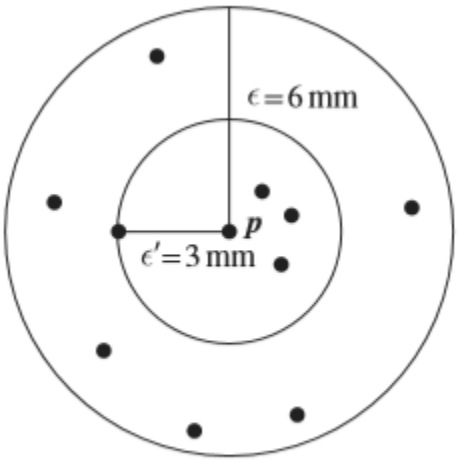
- ❖ Objects in a denser cluster are listed closer to each other in the cluster ordering. This ordering is equivalent to density-based clustering obtained from a wide range of parameter settings.
- ❖ Thus, OPTICS does not require the user to provide a specific density threshold. The cluster ordering can be used to extract basic clustering information(e.g., cluster centers, or arbitrary-shaped clusters), derive the intrinsic clustering structure, as well as provide a visualization of the clustering.

<u>OPTICS needs two important pieces of information per object:</u>

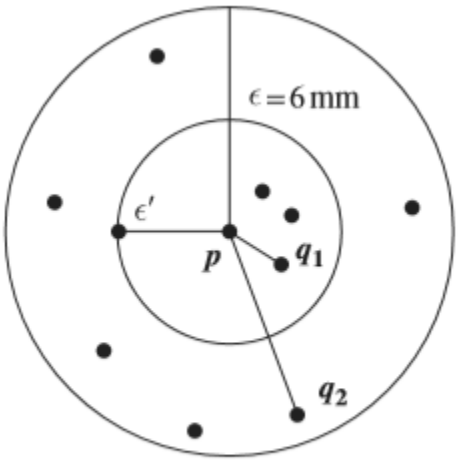
- The core-distance of an object p is the smallest value 0 such that the E-neighborhood of p has at least MinPts objects. That is, 0 is the minimum distance threshold that makes p a core object. If p is not a core object with respect to E and MinPts, the core-distance of p is undefined.
- The reachability-distance to object p from q is the minimum radius value that makes p density-reachable from q. According to the definition of density-reachability, q has to be a core object and p must be in the neighborhood of q. Therefore, the reachability-distance from q to p is max{core-distance(q), dist(p, q)}. If q is not a core object with respect to E and MinPts, the reachability-distance to p from q is undefined.

Core-distance and reachability-distance.

Below Figure illustrates the concepts of coredistance and reachability-distance. Suppose that E =6 mm and MinPts=5. The coredistance of p is the distance, E0, between p and the fourth closest data object from p. Thereachability-distanceofq1 frompisthecore-distanceofp(i.e., E0=3mm)because this is greater than the Euclidean distance from p to q1. The reachability-distance of q2 with respect to p is the Euclidean distance from p to q2 because this is greater than the core-distance of p.



Core-distance of p



Reachability-distance $(p, q_1) = \epsilon' = 3$ mm
Reachability-distance $(p, q_2) = dist\,(p, q_2)$

OPTICS computes an ordering of all objects in a given database and, for each object in the database, stores the core-distance and a suitable reachability-distance. OPTICS maintains a list called Order Seeds to generate the output ordering. Objects in Order

Seeds are sorted by the reachability-distance from their respective closest core objects, that is, by the smallest reachability-distance of each object. OPTICS begins with an arbitrary object from the input database as the current object, p. It retrieves the E-neighborhood of p, determines the core-distance, and sets the reachability-distance to undefined. The current object, p, is then written to output.

- ✓ If p is not a core object, OPTICS simply moves on to the next object in the Order Seeds list (or the input database if Order Seeds is empty).
- ✓ If p is a core object, then for each object, q, in the E-neighborhood of p, OPTICS updates its reachability-distance from p and inserts q into Order Seeds if q has not yet been processed.
- ✓ The iteration continues until the input is fully consumed and OrderSeeds is empty.

The structure of the OPTICS algorithm is very similar to that of DBSCAN. Consequently, the two algorithms have the same time complexity. The complexity is O(nlogn) if a spatial index is used, and O(n2) otherwise, where n is the number of objects.

DENCLUE: Clustering Based on Density Distribution Functions

Density estimation is a core issue in density-based clustering methods. DENCLUE (DENsity-based CLUstEring) is a clustering method based on a set of density distribution functions

- ❖ In probability and statistics, densityestimation is the estimation of an unobservable underlying probability density function based on a set of observed data.
- ❖ Formally, let x1,...,xn be an independent and identically distributed sample of a random variable f. The kernel density approximation of the probability density function is

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

A kernel can be regarded as a function modeling the influence of a sample point within its neighborhood.

- Technically, a kernel K() is a non-negative real-valued integrable function that should satisfy two requirements: R+∞ −∞ K(u)du=1 and K(−u)=K(u) for all values of u.
- A frequently used kernel is a standard Gaussian function with a mean of 0 and a variance of 1:

$$K\left(\frac{x - x_i}{h}\right) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x - x_i)^2}{2h^2}}.$$

A cluster in DENCLUE is a set of density attractors X and a set of input objects C such that each object in C is assigned to a density attractor in X, and there exists a path between every pair of density attractors where the density is above ξ. By using multiple density attractors connected by paths, DENCLUE can find clusters of arbitrary shape.

DENCLUE has several advantages.

- It can be regarded as a generalization of several well-known clustering methods such as single-linkage approaches and DBSCAN.
- Moreover, DENCLUE is invariant against noise.
- The kernel density estimation can effectively reduce the influence of noise by uniformly distributing noise into the input data.

Quality and Validity of Cluster Analysis

a few methods to choose from for measuring the quality of a clustering. these methods can be categorized into two groups according to whether ground truth is available.

Extrinsic Methods

Whether an extrinsic method is effective largely depends on the measure, Q, it uses. In general, a measure Q on clustering quality is effective if it satisfies the following four essential criteria:

Cluster homogeneity

This requires that the more pure the clusters in a clustering are, the better the clustering. Suppose that ground truth says that the objects in a data set, D, can belong to

categories L1,...,Ln. Consider clustering, C1, wherein a cluster C ∈C1 contains objects from two categories Li,Lj (1≤i < j≤n).

Cluster completeness

This is the counterpart of cluster homogeneity. Cluster completeness requires that for a clustering, if any two objects belong to the same category according to ground truth, then they should be assigned to the same cluster.

Rag bag

a "rag bag" category containing objects that cannot be merged with other objects. Such a category is often called "miscellaneous," "other," and so on. The rag bag criterion states that putting a heterogeneous object into a pure cluster should be penalized more than putting it into a rag bag. In other words, C0 in C2 is a rag bag. Then, a clustering quality measure Q respecting the rag bag criterion should give a higher score toC2, that is, Q(C2,Cg) > Q(C1,Cg).

Small cluster preservation

If a small category is split into small pieces in a clustering, those small pieces may likely become noise and thus the small category cannot be discovered from the clustering. The small cluster preservation criterion states that splitting a small category into pieces is more harmful than splitting a large category into pieces.

- Many clustering quality measures satisfy some of these four criteria. Here, we introduce the BCubed precision and recall metrics, which satisfy all four criteria. BCubed evaluates the precision and recall for every object in a clustering on a given data set according to ground truth. The precision of an object indicates how many other objects in the same cluster belong to the same category as the object.
- Formally, let D={o1, ..., on}be a set of objects, and C be a clustering on D. Let L(oi) (1≤i≤n) be the category of oi given by ground truth, and C(oi) be the cluster ID of oi in C. Then, for two objects, oi and oj, (1≤i,j,≤n,i6=j), the correctness of the relation between oi and oj in clustering C is given by

$$\text{Correctness}(o_i, o_j) = \begin{cases} 1 & \text{if } L(o_i) = L(o_j) \Leftrightarrow C(o_i) = C(o_j) \\ 0 & \text{otherwise.} \end{cases}$$

BCubed precision is defined as

$$\text{Precision BCubed} = \frac{\sum_{i=1}^{n} \frac{\sum_{o_j: i \neq j, C(o_i) = C(o_j)} \text{Correctness}(o_i, o_j)}{\|\{o_j | i \neq j, C(o_i) = C(o_j)\}\|}}{n}.$$

BCubed recall is defined as

$$\text{Recall BCubed} = \frac{\sum_{i=1}^{n} \frac{\sum_{o_j: i \neq j, L(o_i) = L(o_j)} \text{Correctness}(o_i, o_j)}{\|\{o_j | i \neq j, L(o_i) = L(o_j)\}\|}}{n}.$$

Intrinsic Methods

Intrinsic methods evaluate a clustering by examining how well the clusters are separated and how compact the clusters are.

- Many intrinsic methods have the advantage of a similarity metric between objects in the data set.
- The silhouette coefficient is such a measure. For a data set, D, of n objects, suppose D is partitioned into k clusters, C1,...,Ck. For each object o ∈D, we calculate a(o) as the average distance between o and all other objects in the cluster to which o belongs. Similarly, b(o) is the minimum average distance from o to all clusters to which o does not belong.
- Formally, suppose o∈Ci (1≤i≤k); then

$$a(o) = \frac{\sum_{o' \in C_i, o \neq o'} dist(o, o')}{|C_i| - 1}$$

and

$$b(o) = \min_{C_j : 1 \leq j \leq k, j \neq i} \left\{ \frac{\sum_{o' \in C_j} dist(o, o')}{|C_j|} \right\}.$$

The **silhouette coefficient** of o is then defined as

$$s(o) = \frac{b(o) - a(o)}{\max\{a(o), b(o)\}}.$$

To measure a cluster's fitness within a clustering, we can compute the average silhouette coefficient value of all objects in the cluster. To measure the quality of a clustering, we can use the average silhouette coefficient value of all objects in the data set. The silhouette coefficient and other intrinsic measures can also be used in the elbow method to heuristically derive the number of clusters in a data set by replacing the sum of within-cluster variances.